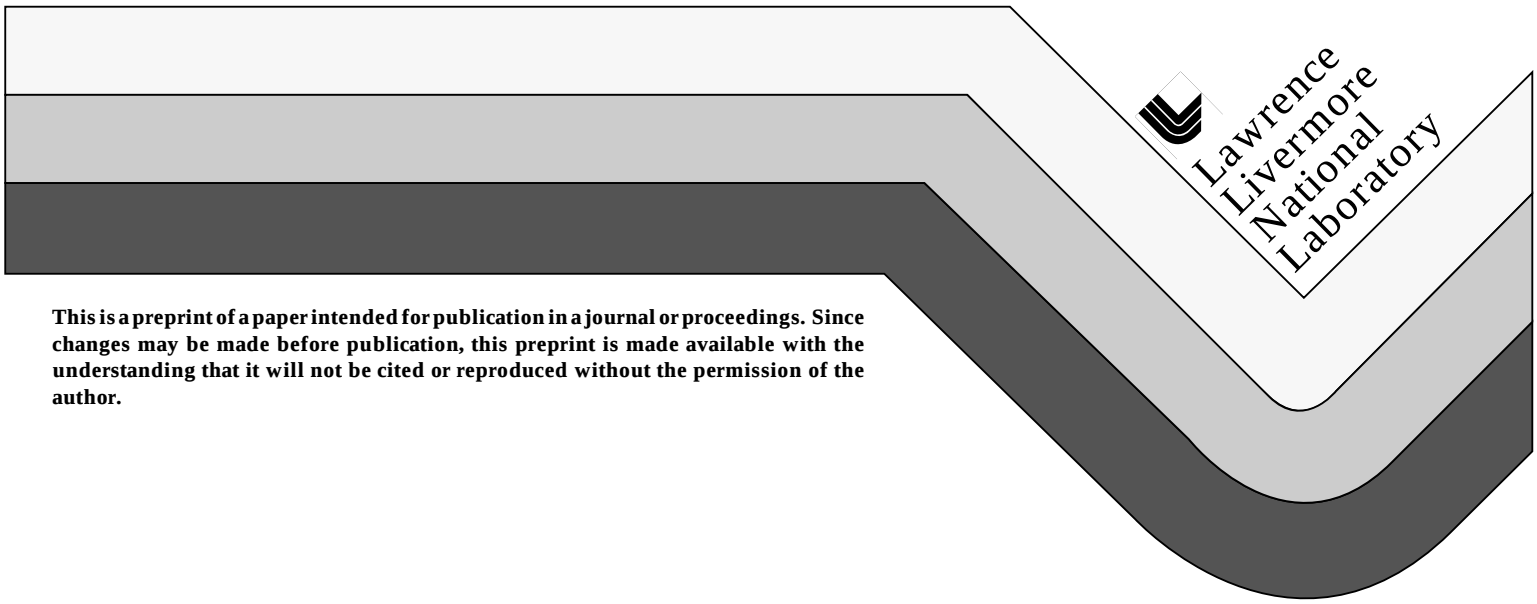
UCRL-JC-121027
PREPRINT

Report of the Hacker Attack Working Group August 1994

Frank Swift

This paper was prepared for submittal to the
1995 DOE Computer Security Group Training Conference
Milwaukee, WI
May 1-4, 1995

February 1995



Report of the Hacker Attack Working Group

August 1994

Members of the Working Group: Richard Feingold, **Dave Grubb (Chair)**, Larry Little, Paul Mauvais, Dave Osterman, Joe Ramus, John Reynolds, Jim Sharp, Larry Snyder, Frank Swift.

The Hacker Attack Working Group was formed to study and recommend specific ways to protect LLNL computers and their assets from attacks from the Internet. The nominal timeline for the study and making this report was four weeks from the date the task was assigned. Accordingly, the group conducted a series of meetings to scope out the problem and possible solutions (see Appendix A). The group performed these functions and reached consensus on a series of recommended solutions. These solutions are listed and described in the body of this report.

The problem we addressed: Hacker attacks from the Internet against computers running UNIX operating systems at LLNL.

Attacks from the Internet - Due to security holes in the operating system(s), the fact that we are using networks (Ethernet) that are the equivalent of telephone party lines where it is easy to listen in, and less than complete system administration, many of the computers attached to Open LabNet are vulnerable to attacks from hackers on the Internet. Our primary concern is with computers running the UNIX operating system. Experience has shown that these machines are the most frequently attacked and, not surprisingly, the most vulnerable to attack. Many of the Lab's Macintosh computers are attached to Open LabNet and are, therefore, at risk. However, today, there are far fewer ways to attack a Macintosh from the Internet. In general, a Macintosh user has to be running a particular piece of software and have it configured in a particular way before his/her Macintosh is vulnerable to Internet attack (see Appendix B). In contrast, due to the security holes in many of the varieties of the UNIX operating system, all a user has to do is have their computer turned on and connected to Open LabNet for it to be vulnerable. For completeness, we note that there are many other computers at the Lab running other operating systems such as VMS. Many of the recommendations we make in this report are applicable and have value to protecting these computers, too.

We also note that our focus was clearly on attacks from outside the Lab. There are also potentially attackers (employees, contractors, etc.) inside the Lab. At this time, we believe the risk from the "insider" threat is much less than the threat from external attacks. Again, though, many of the same actions we recommend to protect against outsider attack will also work against insider attacks.

As part of our process, we discussed a number of different ways of protecting computer systems and the pluses and minuses of each. A summary of these discussions is presented in Appendix C.

We based our recommendations on providing greater and greater degrees of protection to the computer and the data on it.

Recommended Protection Level	Types of computer/data protected
1	All computers (UNIX)
2	Sensitive data such as evaluations and rankings
3	Sensitive data such as Official Use Only, In Strict Confidence and CRADA protected data
4	Sensitive data that the owner wants to protect from snooping as it transverses the net
5	Sensitive data such as UCNI data
6	Data determined by the data owner to be highly sensitive or mission critical

The mapping between Protection Level and type of data protected is only a starting point for determining how much effort, resources, and technology to apply to protecting your data. The relevant DOE Order (1360.2B) and good business practice requires that anyone with sensitive data perform a risk analysis to determine the threats, consequences of data compromise/loss, and resources that they have to apply to protecting their data. One organization may have sensitive data such as Official Use Only or In Strict Confidence, but lack the resources to protect the data at Level 3. This organization may choose to protect the data only at Level 1. Another organization may have data that they must protect according to the terms and conditions of their CRADA. This organization may decide that the consequences of loss or compromise of their CRADA data is so severe that they choose to protect the data at Level 5. In both of these cases, the risk analysis and the thought process that goes into making these decisions should be documented. These decisions should also be reviewed and reassessed on a recurring basis.

The protection levels are summarized below and described in detail in a following section. In each case, we first detail the goal of each protection level (e.g. stopping hackers from exploiting known holes in the UNIX operating system); we then make specific recommendations including specific actions that can be taken to achieve these goals. You may choose to achieve the same goal by using different technologies or administrative controls - the key is to achieve a given level of security.

Six levels of (increasing) protection for data on machines connected to Open LabNet - Summary

Level 1 (essentials) - All systems should do:

Goals -

a. Configure and maintain the system giving rights and privileges to only those users and processes which should have them and giving them no more rights or privileges than they need.
b. Be able to recover stolen or destroyed data
c. Plug the known holes in the operating system.
d. Do not allow hackers to grab the root password as it transports across the net

e. Limit access to your computer to only those machines that should reasonably have access
f. Have the computer under the guidance of someone knowledgeable in computer security
g. Make sure the computer's users are aware of computer security issues

Recommended actions:
a. System under good/defensive system administration[1]
b. Data is backed up to an off-line media (e.g. tape) on a regular basis
c. System is running an up-to-date OS with all security patches[2]
d. Use one time passwords (S/key or a smart card) for root account - Also recommended for VMS systems.
e. Use TCP Wrapper (available on dcsp.llnl.gov)
f. System security is monitored and guided by a trained CSSO
g. Users trained on computer security relevant to their machine(s) and the sensitivity level of the data on them

Level 2:

Goals -
a. Provide Level 1 protection plus
b. Prevent hackers from grabbing user passwords as they transport across the net

Recommended actions -
a. Take all Level 1 actions plus
b. Use one time passwords (S/Key or "Smart Card") for all users - Also recommended for VMS systems.

[1] A good reference book for the security aspects of UNIX system administration is Practical UNIX Security, Simson Garfinkel and Gene Spafford, O'Reily 7 Associates, Inc., 1993

[2] Guaranteed clean and patched OSs to be distributed on CD ROM (inc. encrypted check sums) for major Lab OSs.

Level 3:

Goals:

a. Provide Level 2 protection plus

b. Stop hackers from looking at sensitive data files on disk

Recommended actions -

a. Take all Level 2 actions plus

b. Encrypt all sensitive data files on disk[3] - this is the user's responsibility

Level 4:

Goals -

a. Provide Level 3 protection plus

b. Prevent hackers from snooping information as it transports across the net

Recommended actions -

a. Take all Level 3 actions plus

b. Encrypt all data that goes out onto a network - this is the user's responsibility[4]

Level 5:

Goals -

a. Provide Level 4 protection plus

b. Deny access to the (sub) network supporting the computer(s) with sensitive data

Recommended actions -

a. Take all Level 4 actions plus

b. Isolate machine from Open LabNet with a Firewall and monitor the Firewall computer.

Level 6:

Goal -

a. Guarantee that outside attackers can not access the sensitive data

Recommended actions -

a. Disconnect from Open LabNet

---

[3] Requires a technical working group to recommend products to provide the needed security.

[4] Also requires a working group to identify recommended products/solutions

Descriptions of Recommended Actions:

Level 1 (essentials) - all systems should do:

-- System under good system administration - defensive system administration[5]

The basis of protection from external attacks is good system administration. In this statement we include

-- configuring the system so all files are controlled by users and groups and limiting the number of files and services which are world readable and world write-able

-- having a user tracking system that keeps track of who the users are on the system, why they have accounts (including who authorized the account) and checking all accounts at least once a year to make sure the user still needs the account and is authorized to have it

-- limiting the number of people with root access to the bare minimum level and having individual root accounts

-- limiting the number of setuid and setgid programs and making them non-root setuid if possible

-- making sure everyone has a good (not easily guessed or broken) password

-- turning off all unneeded services (e.g. tftp) or limiting access to these utilities (see TCP Wrapper below)

-- not putting Internet wide services such as gopher, WWW, or anonymous ftp on a machine that stores or processes sensitive data

-- making sure the machine is registered with the Open LabNet registrar (registrar@llnl.gov) including the name of the system administrator, the local network administrator, and the responsible manager or the machine owner

-- backup the system and files regularly and test the backups every so often to make sure you really could recover your data if necessary

-- checking regularly for known vulnerabilities including checking regularly to see if the Ethernet interface has been put in promiscuous mode

-- keeping the operating system patched and up to date (see below.)

We also encourage system administrators to regularly run programs such as Tripwire and COPS or SPI (Garfinkel and Spafford, op. cit.).

Cost: Providing these functions is part of good system administration. The costs incurred are those for training system administrators and for ranking and rewarding the ones who do this difficult job well. There will also be recurring costs associated with requiring system administration on individual user machines if these machines are not already under the care of a systems administrator. Individual users can provide the function of security administration if they have the training, knowledge, and tools to do the administration. The important point is that someone must perform this function. A single poorly or non-administered machine on the net makes all its neighbors more vulnerable to attack.

-- System is running an up-to-date OS with all security patches[6]

---

[5] A good reference book for the security aspects of UNIX system administration is Practical UNIX Security, Simson Garfinkel and Gene Spafford, O'Reily & Associates, Inc., 1993

[6] Guaranteed clean and patched OSs to be distributed on CD ROM (inc. encrypted check sums) for major Lab OSs.

We have found from hard experience that the makers of operating systems tend to put the most effort into providing security fixes for the most recent versions of their operating system (the one they want you to buy). This is the basis for our recommendation that systems run up-to-date versions of the operating system.

However, it is often difficult to know which version of the operating system (OS) is in the best shape from a security point of view and which system patches work with which version of the OS. Accordingly, we recommend that we create for internal (to LLNL) distribution Compact Disks (CDs) with pre-configured and patched versions of the most common operating systems used at LLNL. It would be the task of the OCSSOs working with their CSSOs and system administrators to determine which operating systems will be supported on a Lab-wide basis and how the operating system would be configured on the CDs.

System administrators and knowledgeable users could then install the operating system provided on the CDs and know that they have a system which is clean and as secure as possible.

In addition, we are recommending that the we also maintain an LLNL patch server which will include tested and clean versions of recent patches and installation scripts for installing those patches. Again, only the most common (at LLNL) versions of UNIX will be supported.

Cost:

One time costs:	-- Hardware and software to write CDs: $10k.
Reoccurring costs	-- Staff to configure and maintain the operating system for distribution including adding patches as necessary and writing the CDs: 1/2 - 1 FTE per supported version of UNIX
	-- CDs: $25 per CD
	-- Put computers on maintenance so they can legally have access to operating system updates: $??

-- Use one time passwords (S/key or "Smart Card") <u>for root account</u>

The most recent hacker attacks have proven that having a "good" password is no longer sufficient protection from hacker attacks. A hacker who breaks into a computer on the network and can control the Ethernet interface on that computer can "grab" all passwords as they transport across the net during login. For this reason, we are recommending that all UNIX computers use a "one-time" password mechanism for the root password(s). A one-time password is a password that is only used once. Unlike traditional passwords where the same password is used every time a user logs in into a system, one-time passwords are used once and the same password is never used again. There are two principle ways of generating the one-time password: the first relies on time synchronization between the computer and an electronic card carried by the user. Both the computer and the card calculate what the user's password should be at a given point in time. The user types in the password displayed on the card and sends it to the computer. The computer matches the password the user sent against the one it calculated and if they match it grants the user access to the system. The second method is based on a challenge and a response. As the user logs in, the computer sends the user a series of characters. The user inputs these characters into a local computing device (electronic card, PC, portable computer, etc.) and calculates a response which the user then enters and sends back to the originating computer. If what the user sends back matches the password the originating computer expected, then the user is given access to the system.

Our working group looked at both public domain products and commercially available and supported products. For groups with sufficient resources ($$) we recommend buying the "Smart Card" software and gold cards[7]. Costs are discussed below. For groups who want to do something immediately or can not afford the "Smart Card" solution, we recommend using the public domain software S/Key. Our view of the pluses and minuses of these two solutions are provided in Appendix E.

One Smart card can operate in either time synchronous mode or challenge/response mode. It also supports up to nine administrative domains with one card so it would be possible for a user to have accounts in up to nine different administrative domains (i.e. a system of computers under one management/administrative structure) using only one card.

S/Key is a software-only solution that works on a challenge/response basis. The S/Key software is available via anonymous ftp from the DCSP server (dcsp.llnl.gov.) Included with the software are software packages for Macs and DOS machines which calculate the response to the host machine's challenge. If you are going to be on travel or somewhere where you do not have access to a local computing capability (such as a PowerBook), you can print out a list of valid responses a head of time that you can then enter from a dumb terminal.

Costs: "Smart Card" - A site license for all UNIX machines at LLNL for the "Smart Card" server software costs $180k. We estimate that it will require

[7]Enigma logic is working on a software only solution that eliminates the need for the electronic card. This solution is available today (6/94) for the IBM PC and is promised for Sun workstations and Macintoshes by the end of the summer. They will port the software solution to other platforms if there is sufficient market demand.

approximately 5% of a system administrator's time to administer the one-time password service after it is installed and configured and user accounts are setup. Each user also needs an "Smart Card" Gold card which will cost between $40 - $50 depending on how many cards we buy. Because the batteries in the "Smart Card" card can not be replaced, there is a recurring cost associated with replacing the cards (at another $40 - $50 per user) every three to five years. This is one of the reasons we are interested in a smart card's software only solution which does not require a smart card. We anticipate programming the seeds into the cards ourselves. The hardware costs for the programming device is about $2k. We anticipate that this activity will be performed at one or more of LLNL's major computer centers and that the incremental man-power requirement for this activity will be small.

S/Key - The S/Key software is freely available. We estimate that the incremental cost to maintain the software will be approximately 5% of a system administrator's time. See Appendix F for additional information on S/Key.

-- Use TCP Wrapper (available on dcsp.llnl.gov)

TCP Wrapper is software that is freely available for UNIX systems. It has been referred to as a "poor man's Firewall". TCP wrappers allows the system administrator to selectively reject requests for services such as tftp, exec, ftp, rsh, telnet, rlogin, finger, and others. As an example, it is possible to setup TCP Wrappers on a computer such that, that computer will not accept any tftp (trivial file transfer protocol) requests from outside the domain llnl.gov. As a second example, it is possible to setup TCP Wrappers so that it will not accept any requests for services from a particular host such as a host that has been repeatedly used to attack the Lab in the past.

A five year NASA study showed that a large majority of their attacks came from foreign and educational sites. See Appendix D for more information on TCP Wrapper.

Cost: The TCP Wrapper software is freely available on the DCSP server (dcsp.llnl.gov). There will be some small, incremental cost in system administration to setup and administer TCP Wrapper on a given system, but we have in-house experience with the software and any on-site systems administrator can get help in setting it up.

-- System security is monitored and guided by a trained CSSO

All systems connected to the Internet should be under the purview of a trained Computer Systems Security Officer (CSSO).

Costs: May require some organizations to assign additional personnel as CSSOs.

-- Users and managers trained on computer security relevant to their machine(s) and the sensitivity level of the data on them

One of the keys to improving LLNL security against Internet attack is to alert users and managers to the fact that their computers are attached to the Internet and can be attacked. They then need some simple guidelines (akin to the 13 commandments) that they can follow to improve their security.

We recommend providing them with this information via a booklet that alerts them to the dangers of attack from the Internet and gives them a few, simple steps they can take (such as these recommendations) to protect their date. The booklet would be briefed to the 400 before it is sent out. It would be kept up to data and available on-line. It would also be given out as part of the new employee orientation, as part of the CRADA Principle Investigator's Handbook, and a synopsis would be included in the yearly SAFE security reminder that is sent to all employees yearly.

Cost: It is the joint responsibility of the CS organization and the OCSSOs to develop the information in the booklet. We recommend a focused working group to develop the material for the booklet. We estimate a printing cost (including TID editor support) of $12k. The yearly update (recurring) costs should be comparable ($12k/yr.).

Level 2:

-- Level 1 + one time passwords (S/Key or "Smart Card") <u>for all users</u>

We recommend that everyone who possibly can, setup their systems with Level 2 security. We have found from experience that once a hacker can break into any user's account, it is probable that the hacker can then gain access to root.

Cost: The incremental cost above that of providing one time passwords for the root account (Level 1), is the cost of the additional smart cards at the rate of $40 - $50 per user (assuming a large order of Smart Cards.) Because the batteries in the "Smart Card" card cannot be replaced, there is a recurring cost associated with replacing the cards (at another $40 - $50 per user) every three to five years. This is one of the reasons we are interested in "Smart Card" 's software only solution which does not require a smart card. We anticipate programming the seeds into the cards ourselves. We also anticipate that the activity of programming the cards and distributing them will be performed by one or more of LLNL's major computer centers and can be readily absorbed into their existing staff requirements (i.e. no new FTEs required.)

Level 3:

-- Level 2 + encrypt all sensitive data files on disk - this is the user's responsibility

Encrypting data on a disk is a cost effective way to protect it from both internal (to LLNL) and external (Internet) attacks. The key drawbacks to encrypting data that is stored on disk are:

a. it takes time to encrypt the data on the disk when it is stored and it takes time to decrypt it every time you want to use it.

b. users sometimes forget their encryption key. When this happens the data is lost unless the particular software package allows what is called an administrator's key. This is an encryption key that an administrator can use to decrypt the data even if the user forgets his/her key. We will need a working group to look for products that provide the administrator's key functionality.

Level 1 and Level 2 protections can be implemented by system administrators. Level 3 and Level 4 protection are the user's responsibility. System administrators can install the

necessary software/hardware, but it is the user's responsibility to decide which files need to be protected by encryption.

Cost: The most flexible solution for encryption is to do it in software; the fastest way to do it is in hardware. As noted in the list of levels of protection, we need to have a technical working group look at products that are available and make recommendations. We expect the software will cost between $50 - $200 per user/seat. There will be some small added system administration cost associated with accessing files after users forget their encryption key.

Level 4:

-- Level 3 + encrypt all data that goes out onto a network - this is the user's responsibility

The most recent hacker attacks have shown that hackers are capable of breaking into machines at LLNL and "listening" to the data that goes over sections of Open LabNet. They do this by breaking into a computer and putting the Ethernet card on that computer into what is known as promiscuous mode. Normally, an Ethernet card only picks up the data that is addressed to it. In promiscuous mode, the Ethernet card picks up all the data on the network it is on. In this way hackers can collect all the data (email, QuickMail, file transfer, interactive sessions with remote computers (telnet, SQL access, ...) etc.) on the section of the network where the compromised computer resides.

The best way to protect the data on the network is to encrypt it before it goes out onto the net. Historically we have not done this for a number of reasons including:

1. we had not been compromised in this way before
2. encryption slows down the process of sending data over the net
3. it was hard to distribute the encryption keys.

Number 1 has changed; we have been attacked in a manner which allows the hacker to monitor anything we put out on the net. Number 2 will probably always be true. No matter how fast the hardware and software is at encrypting data, it will always be slower than not encrypting it. The question becomes one of which is worse - the loss of speed or the potential loss of the data. Number 3 is still true, but that will be changing (see below).

Level 1 and Level 2 protections can be implemented by system administrators. Level 3 and Level 4 protection are the user's responsibility. System administrators can install the necessary software/hardware, but it is the user's responsibility to decide which files need to be protected by encryption.

Key distribution is important if person A wants to send encrypted data to person B. Person A has to send person B the encryption key so person B can decrypt the data. The problem has always been how to send person B the encryption key without letting the hacker grab the key as it moved across the net. The generally accepted solution is to have sets of public keys (which are known to everyone) and private keys (which are only known to their owner). The public key/private key method of sending encrypted data is very powerful but it comes at an expense. It requires an infrastructure that hands out the keys to authenticated users and then maintains a database of the keys.

For a Lab wide solution, the other major issues are

1.) using a public key/private key method that runs on all our major computer platforms: Macintosh, IBM PC, and UNIX
2.) using a public key/private key method that is acceptable to DOE.

At the present time, we do not have a Lab-wide solution for key distribution. AIS is setting up the ability to generate and distribute public key/private key pairs for Apple's AOCE environment which is part of the Macintosh operating system 7.1.1 (System 7 Pro). It is unclear whether the keys generated for AOCE could be used on UNIX or PC systems.

The presently most popular public key / private key encryption software for UNIX workstations and PCs is Privacy Enhanced Mail (PEM). Both AOCE and PEM are based on RSA encryption and, therefore, may have the ability to interoperate. Testing will be required to determine whether or not PEM can use keys generated for AOCE and whether or not PEM and AOCE can interoperate.

We recommend that a working group be chartered to look at these issues and make recommendations on creating a Lab-wide solution for creating and distributing public and private encryption keys.

Cost: If we can use the AOCE technology, we anticipate that it will take roughly one FTE to generate and assign the public and private keys and to maintain the key database. LLNL already has a site license of the AOCE client software (part of System 7.1 Pro). The Macintosh AOCE server software costs about $1000 per server. We estimate that we would need roughly as many AOCE servers as we now have QuickMail servers - roughly 50 to 100 servers ($50k -$100k). We anticipate a requirement for 1 man-month of effort to test the AOCE/PEM interoperability. If those test are successful, we could use the PEM software which is freely available for IBM PCs and UNIX workstations.

Level 5:

-- Level 4 + isolate machine from Open LabNet with a Firewall and monitor the Firewall computer.

A Firewall[8] can be an effective deterrent to attacks from the Internet because it restricts who and what kind of network traffic can pass through the Firewall. This working group believes that, if used at all, Firewalls are best applied at the local level (as opposed to the Lab wide level.) The reason is that Firewalls, in addition to restricting who and what can get into your network, also restrict what the valid users of your network can do. Imposing Firewalls at LLNL's connection points to the Internet would place unnecessary restrictions on Internet access by all LLNL users. This working group does not believe that the threat from Internet attack is so great that it warrants this action. Indeed, this working group has serious concerns about using Firewalls in any situation:

-- having a Firewall tends to give people a false sense of security
-- setting up Firewalls around the Lab will tend to isolate the groups with the Firewalls from the rest of the Lab
-- our experience at LLNL has shown that the restrictions placed on users by Firewalls is great enough that the users demand that holes be put in the Firewall to permit key services the users want. The net result is diminished protection from the Firewall.

If a Firewall is needed, we recommend that management consider monitoring the activity on the Firewall. One of the advantages of Firewalls is that they focus attacks on the system to a single point - the Firewall. Added protection can then be added by monitoring the activity on the Firewall for attacks. Network intrusion detection software was developed at LLNL is available for this purpose. Contact Bob Palasek at 422-8527 for more information and a copy of the software.

Cost: Firewalls require both a one-time cost for the computer that functions as the Firewall and a recurring cost of system administration of the Firewall. The system administration costs can very markedly depending upon how many users you are trying to support behind the Firewall and the amount of Internet access you are providing to these users. The least expensive systems are ones where only selected, key resources are protected behind the Firewall. In addition, if the Firewall is monitored, there is a recurring cost to have someone look at and analyze the data that is produced by the monitor.

LLNL has the expertise in-house to setup Firewalls which will provide significantly enhanced protection against Internet attack. We encourage managers to consider employing Firewalls to protect mission critical and highly sensitive systems. Contact the Distributed Computing Support Program (DCSP) for assistance in evaluating and setting up Firewalls.

---

[8]Firewalls and Internet Security - Repelling the Wily Hacker, William R. Cheswick and Steven M. Bellovin, Addison-Wesley, 1994

Level 6:

-- Disconnect from Open LabNet

At a certain point, protection of the data becomes more important than being connected to Open LabNet and the Internet. This is a management decision.

Cost: It has a relatively high cost because it requires not only that you have a computer which is isolated from Open LabNet, but also that the computers which are networked to it are also isolated from Open LabNet.

What Next:

Establishing and maintaining adequate protections from Internet attack is a continuing process. Internet technology is changing rapidly and so are the threats. As part of this study we asked the question - "What will be the next new attack from the Internet ?" . The answers we received fell into three categories:

1. Trojan horses in your Mosaic (and other) applications - Today we take it for granted that when Mosaic launches an application on your machine (such as JPEGviewer) that the application does what we believe it does and nothing more. This is naive at best. We anticipate that we will soon have free and shareware utilities out on the Internet for Mosaic and other applications which will have Trojan horses in them that will steal your password, over write your files, and cause other forms of problems. We will soon need a way to test the applications we get from the Internet for such Trojan horses and to distribute "clean" versions of these programs locally.

2. User identity spoofing - It is already possible to spoof Internet email systems into delivering mail from user 1 at host A and making it look like it came from user 2 at host B. We anticipate that this sort of spoofing will increase. Fortunately encryption technology similar to what we need for encrypting the data we share over the net will also address this problem by providing us with "digital signatures". This is another reason to move ahead on investigating public key/private key encryption.

3. Attacks on routers and domain name servers - Today most routers are vulnerable to many of the same kinds of attacks that UNIX workstations are. The Open LabNet staff is aware of this threat and will be working to protect the backbone routers from attack. Anyone else using or planning to use routers in their local area networks should contact the LabNet staff to discuss threats and possible protection measures. There will also be attacks against domain servers. These attacks will be similar to the ones against routers and workstations.

Appendix A: Hacker Attack Meeting Schedule

Hacker Attack meetings:

Meeting 1 - April 8:

-- Agree on the problem we are solving
-- Brainstorm possible solutions
    - inc. what problem(s) does each solution address

Meeting 2 - April 15:

-- Delineate positives and negatives on each possible solution
-- Determine issues and questions about each solution that need further research
    - someone takes the action to research each questions and problem

Meeting 3 - April 22:

-- Debate the merits and drawbacks for each possible solution
-- Draft a list of preferred solutions

Meeting 4 - April 29:

-- Develop final recommendations

Appendix B: Ways to attack a Macintosh computer from the Internet

For the vast majority of Mac users at LLNL, there is only one way that their Mac can be directly attacked from the Internet - they have to be running the NCSA Telnet (client) application and they have to enable the ftp server function. If a user does this and does not protect the ftp server function by requiring a remote guest to have a user name and password to access the server, then anyone who accesses the user's Mac via ftp has access to all the user's files. The solution is to not enable ftp when running NCSA Telnet. With version 2.5 of Telnet - don't select (put a check mark by) the ftp command in the file menu. With version 2.6 - select the "ftp server" preference and set it to "off." If you are unclear on turning off ftp in NCSA Telnet, read the documentation that is available at NCSA or contact the LLNL Computer Security organization (x2-4655).

Be forewarned that the Macintosh implementation of many of the other Internet server programs including ftp daemon, gopher, WAIS, and the World Wide Web are also potentially subject to Internet attack. Do not place sensitive or mission essential data on a Mac or any other computer which is acting as an Internet server of any kind.

Appendix C: Effectiveness of Security Mechanisms considered, pluses and minuses

Secure Access (one time passwords, Kerberos, s key, smart cards)

+ Reduces password vulnerability
+ Ready for automatic work flow
- Cost, admin., time to integrate in
- not available for all systems
- changes the way you do business (authenticate for every window)

Access Control

+ Denies service
+ Use TCP wrapper to Firewall off services such as denying access from a specific domains, etc.
- Unmanageable if done in routers

Firewalls

+ Forces all remote hosts to act the same way
+ Forces a single point of entry
- performance hit
- lack of flexibility (rules are same for everybody)
- people hide behind a Firewall (false sense of security)
- Admin. load
- Breaks services (requires proxy code)
- Requires committees to decide what comes through
- Only works for IP
- Hackers concentrate on them

Encryption - IP Level, Link Data level (public/private encryption (RSA) eventually must do on net and on disk)

+ widely used
- requires a hierarchy (key management)
- can't route if encrypted below the TCP level

System Administration (defensive administration)

a. software configuration (not world read/write and up-to-date patches

+ required
- hard to do
- have to keep current

b. current OS version

+ vendors support this version and not older ones
+ homogeneity is easier to administer
- not all applications run on latest version

c. enforce good passwords

- multi use passwords don't help

d. audit software (system/user level)

+ helps you see that you're under attack
+ gives you a record of how you were had
- state-of-the-art isn't there yet
- uses a lot of disk space
- performance load

- need to look for exceptions against a base line

e. TCP wrappers, other tools - part of std CD-ROM
+ logs connections/rejects
+ exclude specific domains, machines
- TCP/IP based only
- requires systems administrator effort
- can be Trojaned

f. Sysadmin training/tools
+ must do
+ esp. required for machines with sensitive data
- easy to say, hard to get support from management
- costs time & money

g. CD ROMs (writable) for distributing pre-configured system software
+ verifiable checksums for reference
- requires committees to choose what goes on the CD (per OS)
- have to support multiple OS

h. shadow password file
+ if you've got it - use it

Management:

recognizing importance of
1. personal responsibility
2. Sysadmin
3. CSSO/OCSSO

Network configuration and monitoring:

a. switched nets versus broadcast nets

b. scrambled net boxes 10 BASF -T hub

c. separate white, red, green nets
- can only do to primarily connected buildings
- swing machines
- 500 - 750 $K to connect primary buildings

d. monitoring
- legal issues, requirements
- need to focus on what you're going to look for (e.g. tftp)
- look for promiscuous mode
- put a cron tab file on the OS distribution CD ROM

Users:

user education
+ key part of the total solution-- a must do
+ maybe a requirement for sensitive data access

Other Ideas:

#1 Make all off-site connections come in through one place
+ makes sense
- hard to enforce

- no one organization controls all the access lines

#2 Limit root access
+ especially machines with sensitive data

#3 Should send letter to Ethernet vendors - no promiscuous as mode default

Appendix D: TCP Wrapper

TCP Wrapper Readme -
@(#) README 1.7 92/06/11 22:21:17

General description

With this package you can monitor incoming connections to the SYSTAT, FINGER, FTP, TELNET, RLOGIN, RSH, EXEC, TFTP, TALK, and other IP network services. Connections are reported through the syslog daemon. Requirements are that network daemons are started by the inetd program or something similar, and the availability of a syslog(3) library.

The programs are tiny front ends. By default, they just report the name or address of the remote host and of the requested service, and then invoke the real network daemon; no information is exchanged with the remote client process.

Optional features are: access control that limits the number of hosts that can connect to your network daemons, remote user name lookups with the RFC 931 protocol, and protection against hosts that pretend to have someone else's host name.

The programs can be installed without any changes to existing software or to existing configuration files. Just move the vendor-provided daemons to another directory and install the front ends into their original places. Installation details are given below.

Early versions of the programs were tested with Ultrix >= 2.2, with SunOS >= 3.4 and ISC 2.2. Later versions have been installed on a wide variety of platforms such as SunOS 4.1.1, Ultrix 4.0 and 4.2, Apollo SR10.3.5 and an unknown number of other ones. The software should run without modification on top of most BSD-style TCP/IP implementations.

Restrictions

Some UDP (and RPC) daemons linger around for a while after they have serviced a request, just in case another request comes in. In the inetd configuration file these daemons are registered with the `wait' option. Only the request that started such a daemon will be handled by the front ends. This restriction does not apply to connection-oriented (TCP) services.

Some ConvexOS versions come with a broken recvfrom(2) implementation. This makes it impossible for the daemon front ends to look up the remote host address (and hence, the name) in case of UDP connections.

Access control

When compiled with -DHOSTS_ACCESS, the front-end programs support a

simple form of access control that is based on pattern matching. The access-control software provides hooks for the execution of shell commands when a pattern fires; this feature may be useful to install "booby traps" on specific network services. For details, see the hosts_access(5) manual page, which is in `nroff -man' format.

Detection of hosts that pretend to have someone else's host name

Authentication based on host names, such as used by RLOGIN and RSH, used to work quite reliably when all host name lookups were done from a _local_ hosts file.

With _distributed_ name services, authentication schemes that rely on host names can be subverted by playing games with the address->name and name->address maps that are maintained by some far-away name servers.

The front-end programs verify the remote host name that is returned by the DNS server responsible for the address->name mapping, by looking at the name and address that are returned by the DNS server responsible for the name->address mapping. If any discrepancies are found, the front ends conclude that they are dealing with a host that pretends to have someone else's host name.

If the sources are compiled with -DPARANOID, the front ends will drop the connection in case of a host name/address mismatch. Otherwise, the front ends pretend that the host name is unknown when logging the connection and consulting the optional access control tables.

RFC 931 support

The protocol described in RFC 931 provides a means to get the remote user name from the client host. The requirement is that the client host runs an RFC 931-compliant daemon.

User name lookups are enabled if the source is compiled with -DRFC931. There are some limitations, though: the number of hosts that run an RFC 931 daemon is still small; user name lookups do not work for datagram (UDP) connections. More seriously, remote user name lookups can cause noticeable delays with connections from PCs.

Hooks for extending the access control language

A skeleton is available for adding user-defined extensions to the access control language. The options.c file implements examples that selectively (1) make a daemon front end switch to another user or group id, (2) perform remote user name lookups, and (3) run an alternate server (for example, a bogus ftp daemon that mumbles some faked error message before hanging up).

The language extension hook is not enabled by default because it introduces a minor change to the access control language syntax.

Related software

Versions of rshd and rlogind, hacked to report the remote user name in addition to the remote host name, are available for anonymous ftp (ftp.win.tue.nl:/pub/security/logdaemon.tar.Z). These programs are derived from source code on the first 43BSD network source tape; they have been tested only with SunOS >= 4.0 (the rshd appears to work with Ultrix 4.x, too).

The securelib shared library by William LeFebvre can be used to control access to network daemons that are not run under control of the inetd, such as the RPC daemons that are registered with the portmap daemon. Available as eecs.nwu.edu:/pub/securelib.tar.

An alternative portmap daemon can also help to improve RPC security. ftp.win.tue.nl:/pub/security/portmap.shar.Z was tested with SunOS 4.1.1, Ultrix 3.0 and Ultrix 4.x. The protection is less effective than that of the securelib library because portmap's primary task is to maintain the RPC daemon lookup table. SunOS 4.x users should install the latest revision of the portmap and NIS software.

Another way to manage access to TCP/IP services is illustrated by the servers provided with the authutil package (comp.sources.unix volume 22) by Dan Bernstein. This one relies on RFC 931.

Source for a reasonably fast and portable RFC 931 daemon by Peter Eriksson is available from ftp.lysator.liu.se:/pub/net/pauthd*.tar.Z.

Some TCP/IP implementations come without syslog library. A replacement can be found in ftp.win.tue.nl:/pub/security/surrogate-syslog.tar.Z.

Configuration and installation (the easy way)

An advanced installation recipe is given later on. The "easy" recipe requires no changes to existing software or configuration files.

If you don't run Ultrix, you don't need the miscd front-end program. The miscd daemon implements among others the SYSTAT service, which produces the same output as the WHO command.

By default, the front-end programs assume that the vendor-provided daemons will be moved to the "/usr/etc/..." directory. If you want something else, adjust the REAL_DAEMON_DIR macro in the file tcpd.c (and the REAL_DAEMON macro in miscd.c).

Follow the instructions at the beginning of the Makefile and compile the programs. The result is two binaries (three in case of Ultrix). The `try' program can be used to play with host access control tables and is described in a later section.

The tcpd program can be used for monitoring requests for the telnet,

finger, ftp, exec, rsh, rlogin, tftp, talk, spray, rusers, comsat and
other services that have a one-to-one mapping onto executable files.

Decide which services you want to be monitored. Move the corresponding
vendor-provided daemon programs to the location specified by the
REAL_DAEMON_DIR macro in the file tcpd.c, and copy the tcpd front end
to the locations where the vendor-provided daemons used to be. That is,
one copy of (or link to) the tcpd program for each service that you
want to monitor.

Ultrix only: If you want to monitor and/or restrict access to the
SYSTAT service, move the vendor-provided miscd daemon to the location
specified by the REAL_DAEMON macro in the miscd.c file, and install the
miscd front end into the original miscd location.

Configuration and installation (the advanced way)

Instead of moving the vendor-provided daemons to another directory,
define the REAL_DAEMON_DIR macro in the file tcpd.c (and, if you run
Ultrix, REAL_DAEMON in miscd.c) to reflect the present location of
those daemons. Then follow the instructions in the Makefile and do
a "make".

The tcpd program can be used for monitoring requests for the telnet,
finger, ftp, exec, rsh, rlogin, tftp, talk, spray, rusers, comsat and
other services that have a one-to-one mapping onto executable files.

Install the tcpd command in a suitable place. Apollo UNIX users will
want to install it under a different name because tcpd is the name of
an already existing command. A suitable name for the front-end program
would be "frontd". Then perform the following edits on the inetd
configuration file (usually /etc/inetd.conf):

```
finger  stream  tcp     nowait  nobody  /usr/etc/in.fingerd     in.fingerd
```

becomes:

```
finger  stream  tcp     nowait  nobody  /usr/etc/tcpd           in.fingerd
```

The example applies to SunOS 4.x; other UNIX implementations should not
differ much. Similar changes will be needed for other services that are
to be covered by the tcpd (or frontd) front-end program. Send a SIGHUP
to the inetd process to make the changes effective.

The miscd daemon that comes with Ultrix implements several network
services. It decides what to do by looking at its process name. One of
the services is systat, which is a kind of limited finger service. If
you want to monitor the systat service, install the miscd front end in
a suitable place and update the inetd configuration file:

```
systat  stream  tcp     nowait  /front/ends/miscd          systatd
```

That's right, Ultrix still runs its daemons with root privileges.

Daemons with arbitrary path names

The above tcpd examples work fine with network daemons that live in a common directory, but sometimes that is not possible. You can cope with these cases by specifying, in the inetd configuration file, an absolute path name for the daemon process name. For example,

```
ntalk  dgram  udp    wait   root   /usr/etc/tcpd /usr/local/lib/ntalkd
```

Tcpd ignores the REAL_DAEMON_DIR macro when the daemon process name is an absolute pathname; logging and access control will be based on the last component of the daemon process name.

Testing the access control files

The "try" command, provided with this package, can be used to test out the access control files. The command syntax is:

```
./try process_name hostname        (e.g. ./try in.tftpd localhost)

./try process_name address         (e.g. ./try in.tftpd 127.0.0.1)
```

In the first case, the program will use both the host name and address. In the second case, the program will pretend that the host name is unknown, so that you can simulate what happens when hostname lookup fails. If there are any serious errors they will be reported via the syslog daemon. Run a "tail -f" on the logfile while playing with the "try" command.

Other applications

The access control routines can easily be integrated with other programs. The hosts_access.3 manual page (`nroff -man' format) describes the external interface of the libwrap.a library.

The tcpd front end can even be used to control access to the smtp port. In that case, sendmail should not be run as a stand-alone daemon, but it should be registered in the inetd.conf file. For example:

```
smtp   stream  tcp    nowait  root   /usr/etc/tcpd /usr/lib/sendmail -bs
```

You will periodically want to run sendmail to process queued-up messages. A crontab entry like:

```
0,15,30,45 * * * * /usr/lib/sendmail -q
```

should take care of that.

Acknowledgments

Thanks to Brendan Kehoe (brendan@cs.widener.edu), Heimir Sverrisson (heimir@hafro.is) and Dan Bernstein (brnstnd@kramden.acf.nyu.edu) for feedback on an early release of this product. The host name/address check was suggested by John Kimball (jkimball@src.honeywell.com). Apollo's UNIX environment has some peculiar quirks: Willem-Jan Withagen (wjw@eb.ele.tue.nl), Pieter Schoenmakers (tiggr@es.ele.tue.nl) and Charles S. Fuller (fuller@wccs.psc.edu) provided assistance. Hal R. Brand (BRAND@addvax.llnl.gov) told me how to get the remote IP address in case of datagram-oriented services, and suggested the optional shell command feature. Shabbir Safdar (shabby@mentor.cc.purdue.edu) provided a first version of a much-needed manual page. Granville Boman Goza, IV (gbg@sei.cmu.edu) suggested to use the remote IP address even when the host name is available. Casper H.S. Dik (casper@fwi.uva.nl) provided additional insight into DNS spoofing techniques. The bogus daemon feature was inspired by code from Andrew Macpherson (BNR Europe LTD). I appreciate the code fragments that I received from Howard Chu (hyc@hanauma.jpl.nasa.gov), John P. Rouillard (rouilj@cs.umb.edu) and others. These saved me a lot of work.

Wietse Venema (wietse@wzv.win.tue.nl),
Department of Mathematics and Computing Science,
Eindhoven University of Technology,
The Netherlands.

TCP Wrapper Blurb -
@(#) BLURB 1.5 92/06/11 22:21:40

This package provides a couple of tiny programs that monitor incoming requests for IP services such as TFTP, EXEC, FTP, RSH, TELNET, RLOGIN, FINGER, SYSTAT, and many others.

Optional features are: access control based on pattern matching; remote username lookup using the RFC 931 protocol; protection against rsh and rlogin attacks from hosts that pretend to have someone else's name.

The programs can be installed without requiring any changes to existing software or configuration files. By default, they just log the remote host name and then invoke the real network daemon. No information is exchanged with the remote client process.

Enhancements over the previous release are:

1 - network daemons no longer have to live within a common directory
2 - the access control code now uses both the host address and name
3 - an access control pattern that supports netmasks
4 - additional protection against forged host names
5 - a pattern that matches hosts whose name or address lookup fails
6 - an operator that prevents hosts or services from being matched
7 - optional remote username lookup with the RFC 931 protocol
8 - an optional umask to prevent the creation of world-writable files
9 - hooks for access control language extensions
10 - last but not least, thoroughly revised documentation.

Except for the change described under (2) the present version should be backwards compatible with earlier ones.

Wietse Venema (wietse@wzv.win.tue.nl),
Department of Mathematics and Computing Science,
Eindhoven University of Technology,
The Netherlands.

Appendix E: S/Key pluses and minuses and "Smart Card" pluses and minuses

Both "Smart Card" and S/Key:

+ Far more secure than reuseable passwords
+ Gain a lot of protection for a minimal investment
+ Requires a PIN so are more secure than systems which just use a time synch

- Change the way you work
- Root can su to a user's account if the user has a one time password
- The one time password is new and is changing quickly

"Smart Card":

+ Software works with almost any brand of smart card
+ Available on all major UNIX platforms
+ Vendor supported
+ Software only (smart card not required) version available soon
+ Can operate in either synch or asynch mode (time synchronized or challenge/response)
+ Works on routers

- Until software only version is available, must carry a smart card
- Card cost $40 - $50 per user (in large quantities)
- Software cost roughly $180k for an LLNL site license for UNIX platforms, VMS would be additional cost

S/Key

+ Software is freely available
+ Runs on all Major UNIX platforms
+ Smart cards not required
+ Don't have to worry about time synch'ing - works on challenge/response
+ Can limit when, how many times, and during what time period a user can login
+ Can modify ftp, rsh, etc to require one-time password

- Not vendor supported
- Many different versions available on the Internet (original at AT&T)
- If you do not have a local computer capability, you have to pre-print out a list of usable passwords
- If you loose your list, it is hard/impossible to login until you get a new list or gain access to a local computing capability (like a PowerBook)

Appendix F: Description of The S/KEY One-Time Password System -

Neil M. Haller nmh@thumper.bellcore.com
Philip R. Karn karn@chicago.qualcomm.com

ABSTRACT

The S/KEY one-time password system provides authentication over networks that are subject to eavesdropping/reply attacks. This system has several advantages compared with other one-time or multi-use authentication systems. The user's secret password never crosses the network during login, or when executing other commands requiring authentication such as the UNIX passwd or su commands. No secret information is stored anywhere, including the host being protected, and the underlying algorithm may be (and it fact, is) public knowledge. The remote end of this system can run on any locally available computer. The host end could be integrated into any application requiring authentication.

Trademarks

Attributes of the S/KEY One-Time Password System

The S/KEY authentication system is a simple scheme that protects user passwords against passive attacks. It is not as powerful or general in scope as Kerberos or SDASS; nor does it protect against active attacks. It can, however, be easily and quickly added to almost any UNIX system without requiring any additional hardware and without requiring the system to store information (such as plain text passwords) that would be more sensitive than the encrypted passwords already stored. The S/KEY system can be used with non programmable terminals or personal computers (e.g., systems running DOS or Apple Macintoshes) with conventional communications programs.

Some of the properties of the S/KEY system are:

- Eavesdropping protection
- Conceptually simple and easy to use
- Based on a memorized secret password; does not require a special device although it can easily be adapted to do so.
- Can be automated for authentication from a trusted system.

(Can also be partially automated for fast operation.)

- o No secret algorithms.
- o No secrets stored on host.

Description of the S/KEY One-Time Password System

There are two sides to the operation of our one-time password system. On the user (or client) side, the appropriate one-time password must be generated. On the system (server) side, the one-time password must be verified. One time passwords are generated and verified using a one-way function based on MD4 [Rivest]. (Conversion to MD5 would be trivial)

We have defined our one-way function to take 8 bytes of input and to produce 8 bytes of output. This is done by running the 8 bytes of input through MD4 and then "folding" pairs of bytes in the 16-byte MD4 output down to 8 bytes with exclusive-OR operations. This allows us to apply the one-way function an arbitrary number of times.

Generation of One-Time Passwords

The sequence of one-time passwords is produced by applying the one-way function multiple times. That is, the first one-way password is produced by running the user's secret password (s) through the one-way function some specified number of times, (n). Assuming n=4,

$$p(1) = f(f(f(f(s))))$$

The next one-way password is generated by running the user's password through the one-way function only n-1 times.

$$p(2) = f(f(f(s)))$$

An eavesdropper who has monitored the use of the one-time password p(i) will not be able to generate the next one in the sequence p(i+1) because doing so would require inverting the one-way function. Without knowing the secret key that was the starting point of the function iterations, this can not be done.

Seeding the Password

A user might want to use the same secret password on several machines, or might allow the iteration count to go to zero. An initial step concatenates a seed with the arbitrary length secret password, crunches the result with MD4, and folds the result to 64 bits. The result of this process is then iterated n times.

System Verification of Passwords

The host computer first saves a copy of the one-time password it receives, then it applies the one-way function to it. If the result does not match the copy stored in the system's password file, then the request fails. If they match, then the user's entry in the system password file is updated with the copy of the one-time password that was saved before the final execution (by the server) of the one-way function. This updating advances the password sequence.

Because the number of one-way function iterations executed by the user decreases by one each time, at some point the user must reinitialize the system or be unable to log in again. This is done by executing a special version of the passwd command to start a new sequence of one-time passwords. This operation is essentially identical to a normal authentication, except that the one-time password receive over the network is not checked against the entry already in the password file before it replaces it. In this way, the selection of a new password can be done safely even in the presence of an eavesdropper.

Operation of S/KEY One-Time Password System

Overview

The S/KEY one-time password authentication system uses computation to generate a finite sequence of single-use passwords from a single secret. The security is entirely based on a single secret that is known only to the user. Alternatively, part of or the entire secret can be stored in a non-retrievable way, in the computing device.

Generation of S/KEY One-Time Passwords

As mentioned above, the one-time password sequence is derived from the secret password using a computer. The required computation has been executed on a variety of PC and UNIX class machines including notebook and palm-tops. A vendor has estimated that credit card size devices could be built for less than $30 in large quantities.

The program can also be stored on and executed from a standard floppy disk. This would allow operation on a remote computer that could not be entirely trusted not to contain a Trojan Horse that would attempt to capture the secret password. It is sometimes useful to pre-compute and print several one-time passwords. These could be carried on a trip where public terminals or workstations were available, but no trusted local computation was available.

Description of Operation

The following narrative describes the procedure for logging into a UNIX system using the S/KEY one-time password system. To illustrate the

most complex case, we assume a hand-held PC compatible computer is used.

- The user, call her Sue, identifies herself to the system by login name.

- The system issues a challenge including the sequence number of the one-time password expected and a "seed" that is unique to the system. This "seed" allows Sue to securely use a single secret for several machines. Here the seed is "unix3" and the sequence number is 54.

- Sue enters 54 and unix3 into her palm-top computer. She is prompted for her secret password.

- Sue enters her secret password that may be of any length. The palm-top computes the 54th one-time password and displays it.

- Sue enters the one-time password and is authenticated.

- Next time Sue wants access, she will be prompted for one-time password sequence number 53.

Semi-Automated Operation

The complexity illustrated above is necessary only when using a terminal that is not programmable by the user, or when using a non-trusted terminal. We have built semi-automatic interfaces for clients using communications software on popular personal computers. The following example illustrates logging in using a trusted personal computer and a popular terminal emulation program.

- Before starting the communication program, Sue runs the CTKEY program that ties a TSR to a "hot-key" such as F10.

- Sue identifies herself by login name as above.

- The system issues the same challenge including the seed "unix3" and the sequence number 54. The host system now expects an s/key one-time password.

- Sue presses the hot-key and is then prompted for a secret password by the TSR program on the local system.

- In response to Sue's secret password, the 54th one-time password is displayed at the position of the cursor.

- Sue presses "Insert" and the terminal emulator transmits the one-time password completing the authentication.

If the personal computer were in a trusted location, an option of the CTKEY program allows the secret password to be stored in a local file.

Form of Password

Internally the one-time password is a 64 bit number. Entering a 64 bit number is not a pleasant task. The one-time password is therefore converted to a sequence of six short words (1 to 4 letters). Each word is chosen from a dictionary of 2048 words. The contents of this dictionary is not a secret.

Source Screening

It is frequently desirable to allow internal access with a multi-use password while requiring one-time passwords for external access. A screening table provides this function. When this table is present, login attempts that pass the screening test are permitted to use the normal password or a one-time password. Others are notified that the use of the one-time password is required.

Password echo

Normally systems disable printing during the typing of a password so that an onlooker cannot steal the password. With a one-time password, this is unnecessary. The replacement login command allows the user to turn echo on by pressing "return" at the password prompt. This makes it easier to enter the longer one-time password.

Acknowledgments

The idea behind our system was originally described by Leslie Lamport. Some details of the design were contributed by John S. Walden who wrote the initial version of the client software. This work was performed under the auspices of the U.S. Dept. of Energy at LLNL under contract no. W-7405-Eng-48.

References

Eugene H. Spafford, "The Internet worm program: An analysis." Computer Communications Review 19(1):17-57, January 1989.

D. C. Feldmeier and P. R. Karn, "UNIX Password Security - Ten Years Later", Crypto '89 Conference , Santa Barbara, CA August 20-24, 1989.

J. G. Steiner, C. Neuman, and J. I. Schiller. "Kerberos: An authentication service for open network systems." USENIX Conference Proceedings, pp. 191-202, Dallas, Texas, February 1988.

Catherine R. Avril and Ronald L. Orcutt. Athena: MIT's Once and Future Distributed Computing Project. Information Technology Quarterly , Fall 1990, pp. 4-11.

R. L. Rivest, The MD4 Message Digest Algorithm, Crypto '90 Abstracts (August 1990), 281-291.

Leslie Lamport, "Password Authentication with Insecure Communication", Communications of the ACM 24.11 (November 1981), 770-772.

Technical Information Department • **Lawrence Livermore National Laboratory**
University of California • Livermore, California 94551